REMARKS

Status of the Claims

- Claims 1-6 and 8-23 are pending in the Application.
- Claims 1-6 and 8-23 are rejected by Examiner.
- Claims 1, 11, 17, and 21 are amended.

Claim Rejections Pursuant to 35 U.S.C. §112

Claims 11 and 21 are rejected under 35 U.S.C. §112, second paragraph, as being indefinite because of lack of antecedent basis of the term "semantic representation".

Claims 11 and 21 are amended to replace the term "semantic" with "intermediate language" to accommodate antecedent basis. Support for this amendment can be found in paragraphs 0021 and 0046 of the as-filed application. Applicant respectfully requests reconsideration and withdrawal of the 35 USC §112 rejection in light of the amendments to Claims 11 and 21.

Claim Rejections Pursuant to 35 U.S.C. §101

Claims 17 is rejected under 35 U.S.C. §101 as being directed to non-statutory subject matter.

Claims 17-20 are amended to include the term computer-readable storage media, which is defined in the as-filed specification paragraphs 0031-0032 as a tangible element. As such, amended Claims 17-20 are now directed to a computer-readable storage medium which is a tangible article of manufacture under 35 USC §101. Applicant respectfully requests reconsideration and withdrawal of the 35 USC §101 rejection in light of the amendments to Claims 17-20.

Claim Rejections Pursuant to 35 U.S.C. §103

Claims 1-6 and 8-23 stand rejected under 35 U.S.C. §103(a) as being unpatentable under U.S. Pat. No. 7,120,645 to Manikutty et al. (Manikutty) in view of a publication by Shanmugasundaram et al. (Shanmugasundaram) entitled "Relational Databases for Querying

XML Documents: Limitations and Opportunities", Published Sept. 1999 in "Proceedings of the Twenty-Fifth International Conference on Very Large Data Bases", Pages 302-314. Applicant respectfully traverse the rejection.

Claim 1 is amended to include the aspect that "the semantic representation decouples front-end language compilers from back-end query engines that use the semantic representation, such that, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations." Support for this amendment may be found viewing the compilers depicted in as-filed Figure 3 as well as the back-end search engines of as-filed Figure 3 in conjunction with paragraph 0023.

Manikutty at col. 5, lines 13-25 teaches a technique for executing database commands that involve operations on XML constructs includes receiving the database command. It is then determined whether an XML component operation in the database command can be transformed to a relational database operation, which operates on a particular set of one or more relational database constructs, and which does not involve the XML component operation. If it is determined that the XML operation can be transformed, then the XML component operation is rewritten to a particular relational database operation that does not involve the XML component operation. The particular relational database operation on the particular set of one or more relational database constructs is evaluated. (see col. 5, lines 13-25).

Figure 3 of Manikutty is a flow diagram that illustrates a method of rewriting a database query that has an XML operation. (see col. 4, lines 52-54). Figure 4 of Manikutty is a flow diagram of step 320 of Figure 3. Although Manikutty teaches the rewrite of an XML query into SQL, Applicant respectfully submits that Manikutty is absent a teaching of the aspect that utilization of the intermediate language representation, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations as recited in amended Claims 1, 11, 17, and 21.

Pages 9, 12, and 15 of the present Office Action dated 2/7/8 indicates that Manikutty teaches these elements in col. 4, lines 36-62, col. 9, lines 59 to col. 10 line 24, col. 21 line 48 to col. 22, line 29, "wherein the results of the XML generating sub-query are expanded into a second tree of XML operations, the front end language being XML and the back end being XPath traversals". Applicant respectfully disagrees with the conclusion of the Office Action.

Manikutty at col. 4, lines 36-62 teaches:

"According to another aspect of the invention, a technique for executing database commands that involve generating an XML type from relational database constructs includes determining a primitive set of XML generation operations for generating any XML construct from any relational database construct. It is determined whether a database command involves a particular XML generation operation that is not a member of the primitive set. If it is determined that the database command involves the particular XML generation operation that is not a member of the primitive set, then a particular set of one or more XML generation operations from the primitive set is determined that produces the same result as the particular XML generation operation. The particular set of one or more primitive XML generation operations is used instead of the non-primitive operation to evaluate the database command.

This technique allows a DBMS to employ simpler rules for determining when XML components contribute nothing to the outcome of a database command or when XML component operations can be replaced by SQL operations. The technique therefore allows the DBMS to more easily determine when to avoid generating excess XML elements that have to be parsed in an XML processor and when to replace XML operations with SQL operations that can be executed more efficiently in a database. Avoiding such excess generation and parsing and using more SQL operations can often greatly enhance database performance." (Manikutty at col. 4, lines 36-62)

Yet, Applicant notes that no mention is made of the pending amended claim elements "wherein the semantic representation decouples front-end language compilers from back-end query engines that use the semantic representation, such that, when used in a compiler system

having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations."

Manikutty at col. 9, lines 59 to col. 10 line 24 teaches:

"Under some conditions, when the SQL/XML query is being compiled, the SQL/XML DBMS can determine one or more columns of one or more tables where the queried data reside, which columns represent less than the whole XML construct in the tables. When this occurs, the SQL/XML query is rewritten as an SQL query on the one or more columns of the underlying tables where the queried XML data are stored. For example, XPath queries over schema-based and non-schema-based XML type views are rewritten to go directly over the underlying object-relational data. The entire XML construct does not need to be manifested, and, in many cases, the query is rewritten to go directly on relational data. The rewritten query allows efficient access of the relational data, and can lead to orders of magnitude performance gains over previous approaches. Relational indexes can be created and optimized and used for highly efficient query execution. Furthermore, existing relational indexes already created can be used for a new domain of applications, namely, those that use XPath retrieval. In addition, knowledge about the metadata of the underlying data can lead to more efficient access paths, as well as more optimal query execution algorithms

Murthy-1. reveals a mechanism to rewrite a query with an XPath operator, directed to an XML type table, as an SQL query. When the re-written query is executed, some information of interest comes directly from the tables without generating an XML representation and without subsequent parsing by the XPath operation. In some cases, no XML operation need be invoked at all.

In other embodiments disclosed herein, techniques are employed that rewrite a query that includes a sub-query with XML generation operations, either explicitly, or implicitly as a view of XML type."
(Manikutty at col. 9, lines 59 to col. 10 line 24)

Yet, Applicant notes that no mention is made of the pending amended claim elements "wherein the semantic representation decouples front-end language compilers from back-end query engines that use the semantic representation, such that, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations."

Manikutty at col. 21 line 48 to col. 22, line 47 teaches:

"In step 460, XML component operations that operate on the results of the XML generating sub-query are expanded to a second tree of XML component operations. For example, an XPath expression consists of a series of steps, applied in order. The result of one step can be viewed as the input to the next. This leads to the mapping of an XPath expression to a tree of XPath operators. The XPath expression is broken into its component steps. Each step is represented as a subtree using the fundamental XPath operators, plus other operators that cannot be further simplified. The root of this subtree becomes the child of the next set of operators corresponding to the next step. The set of fundamental XPath operators includes the following: XPATG (XPath Attribute get): This corresponds to the step `/child:name`, abbreviated as `/name`. The XPATG operator takes a relative XPath name as well as a context., e.g. /name is converted to XPATG(`name`, `/`); `/lineitem/name` is converted to XPATG(`lineitem`, `/`) under_XPATG(`name`, `/lineitem`); `/*` is converted to XPATG(`*`, `/`). XPLIT (XPath literal): This corresponds to an XPath literal such as `Ashanti`. This is represented as XPLIT(`Ashanti`) with no operands. XPOPR (XPath operator): This corresponds to an operator such as `=`, `+`, `-` etc. For example, `li_msrp<2*li_listprice` results in XPOPR(`<`) with two children. The first operand is XPATG(`li_msrp`) and the second operand is XPOPR(`*`). The latter XPOPR( ) has two children--XPLIT(`2`) and XPATG(`li_listprice`, <context>). XPIDX (XPath Index): This corresponds to the use of indexing, e.g. `/lineitems[1]` would result in an XPIDX(1) over XPATG(`lineitems`). XPPRED (XPath Predicate): This corresponds to the use of predicates, e.g. `/lineitem[li_msrp<2*li_listprice]` results in XPPRED( ) with two children. The first child is XPATG(`lineitem`). The second child is an XPOPR, whose subtree corresponds to `[li_msrp<2*li_listprice]`, i.e. XPOPR(`<`, XPATG(`li_msrp`, `lineitem`), XPOPR(`*`, XPLIT(`2`), XPATG(`li_listprice`, `lineitem`)). At the end of the expansion, the XPath

expression has been converted to a tree of XPath operators, whose net effect is the same as the original XPath expression.

For example, the XPath expression `/PurchaseOrder/Address/City[Zip=94404]` can be broken down into: XPATG(`PurchaseOrder`, `/`) under XPATG(`Address`, `/`) under XPATG(`City`, `/`) under XPPRED. The other child of XPPRED is an XPOPR(`=`). The XPOPR has two children: XPATG(`Zip`, `City`) and XPLIT(94404).
Elimination of Nodes on Two Trees
In step 470, the root node of the first normalized XML generation tree is used as input to a leaf node of the second tree of XML component operations on the sub-query; and it is determined whether and which nodes of the second tree eliminate a node of the first tree. Both nodes are eliminated if the two nodes represent inverse transformations or if the nodes can be replaced by an SQL operation. A node on the XML generation tree is eliminated if the current remaining XPath node does not yield a result from operating on that node of the XML generation tree. If at least one set of nodes can be eliminated, then the query can be rewritten without the node or with the corresponding SQL operation, and control passes to step 494 to rewrite the query during step 340. If no nodes, or too few nodes, can be eliminated, then control passes to step 492 to bypass rewriting of the query."
(Manikutty at col. 21 line 48 to col. 22, line 47)

Yet, Applicant notes that no mention is made of the pending amended claim elements "wherein the semantic representation decouples front-end language compilers from back-end query engines that use the semantic representation, such that, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations."

Accordingly, Manikutty fails to teach at least the elements of "wherein the semantic representation decouples front-end language compilers from back-end query engines that use the semantic representation, such that, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations." at any of the locations cited in the present Office Action on page 9.

Shanmugasundaram teaches the development of algorithms and implementation of a prototype system that converts XML documents to relational; tuples, translates semi-structured queries over XML documents to SQL queries over tables, and converts the results to XML. (See Shanmugasundaram, Abstract).

However, Shanmugasundaram, like Manikutty, fails to teach at least the elements of "wherein the semantic representation decouples front-end language compilers from back-end query engines that use the semantic representation, such that, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations" as indicated in the amended pending claims.

Since the combination of Manikutty and Shanmugasundaram fail to teach or suggest every element of the pending amended independent Claims 1, 11, 17, and 21, then the combination cannot render obvious the pending independent and dependent claims under 35 USC §103(a) via MPEP §2143.03.

Applicant therefore respectfully requests withdrawal of the 35 USC §103(a) rejection and submits that the pending claims patentably define over the cited art because all elements of the independent Claims 1, 11, 17 and 21 are not found in the cited art.

Conclusion

Applicant respectfully requests reconsideration and continued examination of all pending claims in light of the amendment and discussion above. Applicant respectfully submits that all pending claims patentably define over the cited art and respectfully requests a Notice of Allowance for all pending claims.

Respectfully Submitted,

Date: June 6, 2008

/Jerome G. Schaefer/

Jerome G. Schaefer
Registration No. 50,800

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439